To appear in the *International Journal of Production Research*
Vol. 00, No. 00, 00 Month 20XX, 1–17

Order picking in parallel-aisle warehouses with multiple blocks: Complexity and a graph theory-based heuristic

Melih Çelik* and Haldun Süral

Department of Industrial Engineering, Middle East Technical University, Ankara, Turkey

(*Received 00 Month 20XX; accepted 00 Month 20XX*)

In this paper, we consider the order picking problem (OPP), which constitutes one of the special cases of the Steiner traveling salesperson problem and addresses the costliest operation in a warehouse. Given a list of items to be picked and their locations in the warehouse layout, the OPP aims to find the shortest route that starts from a depot point, picks all the items in the list, and returns to the depot. This paper fills two important gaps regarding the OPP. First, to the best of our knowledge, we present the first complexity results on the problem. Second, we propose a heuristic approach that makes use of its graph theoretic properties. Computational experiments on randomly generated instances show that the heuristic not only outperforms its state-of-the-art counterparts in the literature, but it is also robust in terms of changing problem parameters.

Keywords: warehouse logistics; order picking; picker routing; computational complexity; heuristics; graph theory

1. Introduction

With its strong implications on the theory of combinatorial optimization as well as its relevance to numerous real-life decision problems, the *traveling salesperson problem* (TSP) is one of the most extensively studied problems in the literature. Given a complete graph, the TSP seeks to find the Hamiltonian tour with the smallest total edge weight. The *Steiner traveling salesperson problem* (StSP) generalizes the TSP by requiring a subset of nodes to be visited at least once, whereas the remaining nodes may not be visited at all. Whereas both problems are $\mathcal{NP}$−hard in general (e.g., Johnson and Papadimitriou 1985; Cornuéjols, Fonlupt, and Naddef 1985), many polynomially solvable special cases exist as well (Burkard et al. 1998).

This paper focuses on a special case of the StSP, namely the *order picking problem* (OPP) in parallel-aisle warehouses. In a warehouse, order picking refers to the activity of collecting items corresponding to an order or a set of orders (Bartholdi and Hackman 2016). Of the four main activities in a warehouse (receiving, put-away, order picking, and shipping), order picking is the most expensive activity, constituting around 55% of all warehousing costs (Tompkins et al. 2010).

There are various decisions that have to be made in order to perform efficient and effective order picking, including tactical decisions such as assignment of products to the storage areas and zoning of picking areas; as well as operational decisions such as batching of orders, routing of order pickers, and accumulation/sorting of orders (De Koster, Le-Duc, and Roodbergen 2007). For a conventional system in which order pickers need to visit item locations to perform the picks (picker-to-part systems), almost half of the picking time is occupied by traveling between item locations. Therefore, providing efficient picker routing methods can not only improve the response time to an order, but can also decrease the warehousing costs. Motivated by this, the OPP aims to pick and prepare the items corresponding to an order in the shortest amount possible. The objective is to determine the route that minimizes the travel time required for this operation.

*Corresponding author. Email: cmelih@metu.edu.tr

The OPP has received a considerable amount of attention in the literature. There exist polynomial time algorithms for special cases (Ratliff and Rosenthal 1983; Roodbergen and de Koster 2001b) as well as various heuristic procedures (e.g., Hall 1993; Vaughan and Petersen 1999; Roodbergen and de Koster 2001a; Theys et al. 2010). To the best of our knowledge, while special cases have been shown to be tractable, no analysis exists on the computational complexity of the OPP in general, a gap this paper aims to fill. Furthermore, the heuristics particularly developed for the OPP either result in high optimality gaps, or perform well only for a certain set of instances. In this study, we propose a heuristic procedure that is not only robust in terms of the problem settings, but also makes use of the graph theoretic properties of the problem. We test the performance of the heuristic on randomly generated instances, and compare the results with those obtained by its counterparts in the literature.

The remainder of this paper is organized as follows: §2 gives a formal definition of the OPP in parallel-aisle warehouses, describes the graph structure corresponding to it, and provides a summary of the literature on exact and heuristic solution approaches for the problem. §3 discusses the exact solution approaches in greater detail, as they form the baseline for the proposed heuristic and point to the polynomially solvable cases. In §4, complexity results for the OPP are presented, whereas §5 discusses the proposed heuristic approach, called *merge-and-reach*. Computational experiments are presented in §6, and the paper is concluded in §7.

2. Problem Definition and Related Literature

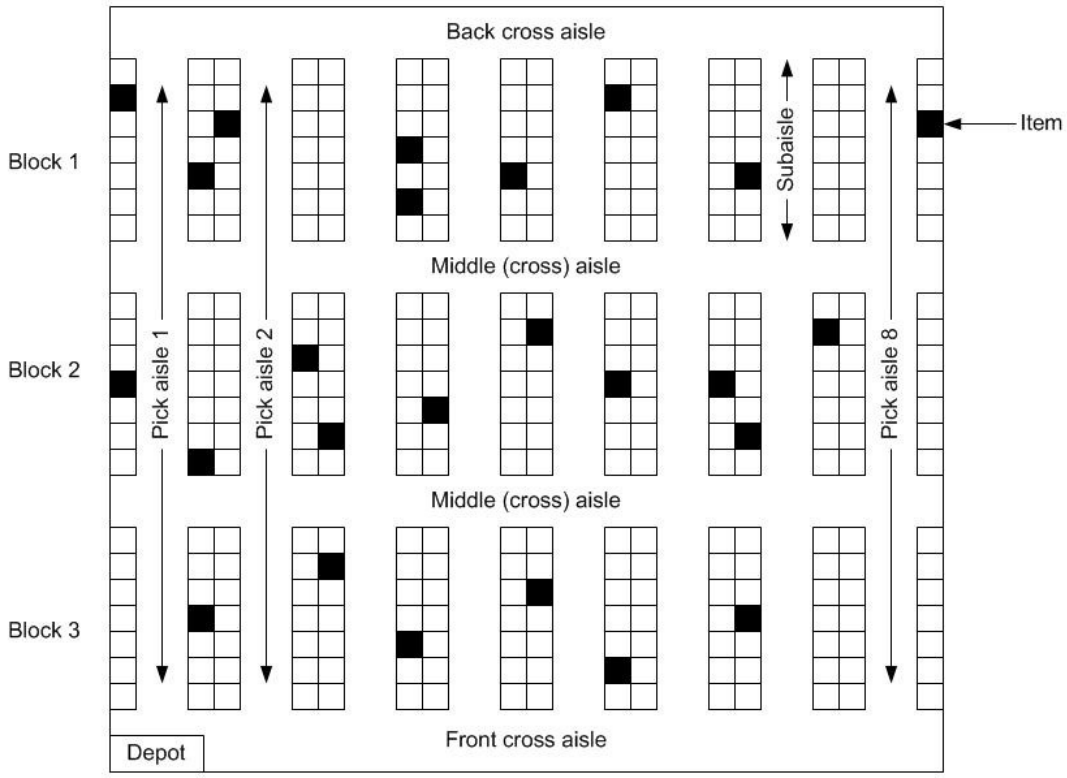
The OPP arises each time an *order* is generated in a warehouse. Given a list of items to be picked (a *pick list*) along with their locations in the warehouse, an uncapacitated order picker needs to visit all corresponding item locations and pick the items.

We focus on the OPP in parallel-aisle warehouses, in which the items are located on pick aisles that are parallel to each other. An example of a parallel-aisle warehouse is shown in Figure 1(a), consisting of eight *pick aisles* that contain the items to be picked (shown as black squares in the figure). The warehouse also contains four *cross aisles*, which are orthogonal to the pick aisles. It is assumed that cross aisles do not contain any pick items. The two cross aisles at the ends of the warehouse are named as *front* and *back cross aisles*, whereas the remaining cross aisles are referred to as *middle aisles*. In the existence of middle aisles, pick aisles are divided into *subaisles* and the warehouse is divided into *blocks*.

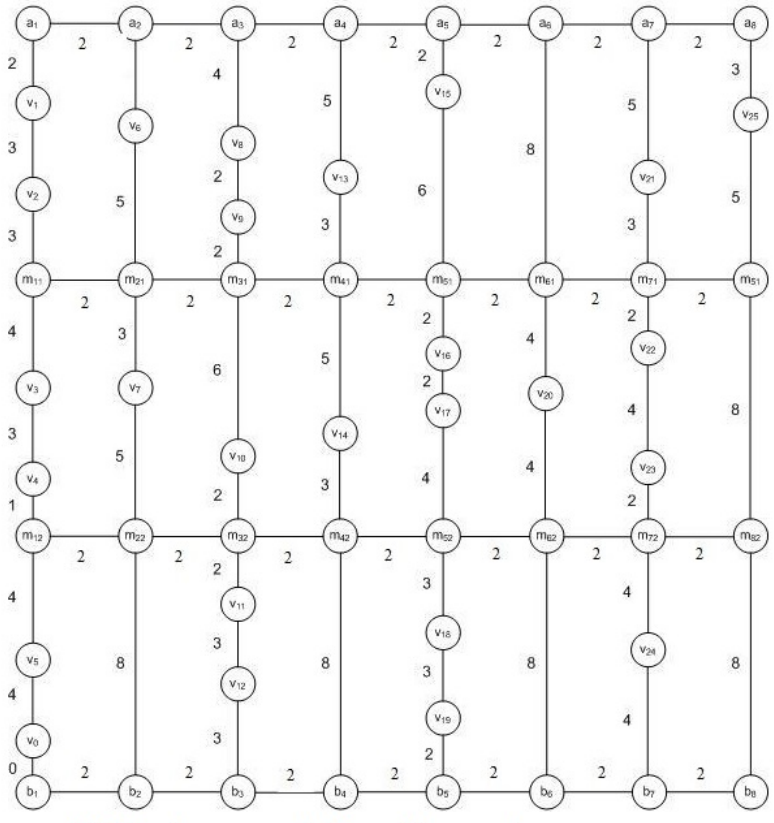
The OPP aims to find a route in which the picker starts at the *depot*, picks all items and returns to the depot in the minimum possible time. Since the time required for acceleration/deceleration and picking the items can be assumed to be constant, the OPP solely focuses on the travel time from an item location (or depot) to another. It is also assumed that the pick aisles are narrow enough to allow for picking from both sides of the aisle without additional time. We denote the OPP with $k-1$ blocks (or $k-2$ middle aisles) as the k-OPP.

The OPP can be defined on a graph $G = (V, E)$, where the vertex set represents the depot, pick items, and corners of cross aisles, and pick aisles. Each item i is denoted as vertex v_i, where v_0 denotes the depot. Corners of the back cross aisle with pick aisle i are denoted by vertex a_i, whereas corners of the front cross aisle with pick aisle i are denoted by vertex b_i. Corners of pick aisle i with middle aisle j (with $j = 1$ being the closest to the back cross aisle and so on) are denoted by vertex m_{ij}. The edges represent the accessibility between items, from a corner to an item and vice versa. Each edge $(i, j) \in E$ has a travel time of w_{ij} associated with it. Figure 1(b) shows the graph representation of the warehouse illustrated in Figure 1(a). Based on this graph definition, the OPP constitutes a special case of the StSP, where v_i represent the nodes that need to be visited and the remaining nodes may not be visited at all (Steiner nodes).

Special cases of the OPP have been shown to be polynomially solvable. A dynamic programming based algorithm is presented by Ratliff and Rosenthal (1983), which is extended to the case of decentralized depositing by De Koster and van der Poort (1998), and to that of the 3-OPP by Roodbergen and de Koster (2001b). The particular cases of 2-OPP and 3-OPP are further discussed in §3. Scholz et al. (2016) formulate a mixed integer programming model that takes into account the special graph structure for the 2-OPP, which



(a) A parallel-aisle warehouse with three blocks, two middle aisles, and 25 pick items.



(b) Graph representation of the warehouse.

Figure 1. Conversion of a 4-OPP instance with 25 items to its graph representation.

is further extended to multiple blocks by Ruberg and Scholz (2016), Scholz (2016), and Pansart, Catusse, and Cambazard (2017). Although the formulations perform better than those for the general TSP or StSP, they can only solve instances of small size, and are easily outperformed by the dynamic programming based approaches.

Despite the tractability of special cases, the optimal tours are generally complicated and hard-to-apply in practice. Consequently, a number of heuristics are proposed in the literature for the OPP. Among these, Makris and Giakoumakis (2003) extend the k-opt heuristic for the 2-OPP, whereas Theys et al. (2010) make use of the Lin-Kernighan-Helsgaun heuristic (Helsgaun 2000) for the more general case. Similarly, De Santis et al. (2018) use an ant-colony optimization method that incorporates the Floyd-Marshall procedure. These approaches directly make use of heuristics for the TSP, and therefore do not take into account the special structure of the OPP graph.

More recently, the literature on order picking considers the picker routing problem along with other tactical or operational-level problems in the warehouse. These involve incorporation of multiple order pickers with congestion (Chen et al. 2013), involvement of human factors in the picking process (Grosse et al. 2015), consideration of weight, fragility, and category constraints into pick sequencing (Chabot et al. 2017), integration of all four basic warehouse functions (Altarazi and Ammouri 2018), joint decisions of picker routing and order batching (Matusiak et al. 2014; Cheng et al. 2015; Öncan 2015; Lin et al. 2016; Chen, Wei, and Wang 2017; Li, Huang, and Dai 2017; Giannikas et al. 2017; Valle, Beasley, and da Cunha 2017), consideration of the assignment of stock keeping units to storage locations (storage assignment) and picker routing simultaneously (Shqair, Altarazi, and Al-Shihabi 2014; Roodbergen, Vis, and Taylor 2015; Dijkstra and Roodbergen 2017; Wu et al. 2017), zone assignment to multiple pickers along with their routing (Matthews and Visagie 2013; Chen et al. 2016; Wu et al. 2017; Kou, Xu, and Yi 2018), simultaneous batching, storage assignment, and picker routing (Chackelson et al. 2013; Scholz and Wäscher 2017), and simultaneous modeling of zoning, batching, and picker routing (Henn and Schmid 2013; Chen et al. 2015). For an extensive survey of order picking systems where planning problems are combined, the interested reader is referred to Gils et al. (2018). In such problems, although a more integrated system is taken into account, picker routing tends to be the easier of the problems, and hence more attention is given to the solution of the other problem(s). Furthermore, as in the stream of studies that focuses solely on picker routing, no attention is given to the special graph structure when considering the routing decisions.

For the 2-OPP, the most commonly used heuristics are summarized in Hall (1993), which include the *S-shape* (traversal), *midpoint*, and *largest gap* heuristics. Starting from the left-most nonempty aisle, the S-shape heuristic completely traverses each aisle with items (except possibly the last one when odd nonempty aisles are present) and returns to the depot. The largest gap heuristic avoids the longest gap between any pair of item nodes or between a corner and item node.

The 2-OPP heuristics are extended to the more general case by Roodbergen and de Koster (2001a), where the single-block versions are applied in sequence on a block-by-block basis, starting from Block 1, proceeding with Block 2, and so on. Vaughan and Petersen (1999) propose the dynamic programming based *aisle-by-aisle* heuristic, which visits each nonempty pick aisle exactly once. Pick aisles constitute the stages and in each stage of the algorithm, for each cross aisle i, one needs to determine the time required to start at the depot, pick all the items in the pick aisles up to i, and exit aisle i from cross aisle j. The optimal solution is the one that picks all the items and ends at the front end of the right-most pick aisle. The *combined* heuristic by Roodbergen and de Koster (2001a) can be considered an extension of the aisle-by-aisle approach, where the dynamic programming algorithm is applied for each block in sequence, starting from Block 1. The heuristic is further improved by making sure that the last block is traversed from right to left, and avoiding the traversal of empty aisles before reaching the first block, culminating in the combined$^+$ heuristic. Computational experiments show that combined$^+$ significantly outperforms the others in 74 of the 80 settings. However, it fails to find good solutions when the number of aisles is large, aisles are shorter and the number of items is higher.

For the OPP instance in Figure 1, Figure 2 shows the resulting S-shape, largest gap, aisle-by-aisle, and combined heuristic solutions, with total travel times of 192, 190, 190, and 182 units, respectively.

Despite the abundance of exact and heuristic solution approaches, the complexity of the problem has

been left open in the literature. Furthermore, state-of-the-art heuristics for the OPP tend to result in significant optimality gaps for at least a subset of instance settings (Roodbergen and de Koster 2001a). Throughout the remainder of this paper, we aim to bridge these two gaps.

3. Exact Solution Approaches for the OPP

The smallest nontrivial cases of the OPP, namely the 2-OPP and 3-OPP, have been shown to be polynomially solvable. Since the heuristic approach in §5 makes use of these, their overviews are provided in this section.

3.1 *2-OPP*

Ratliff and Rosenthal (1983) provide a dynamic programming-based solution algorithm for the 2-OPP, with the aisles constituting the stages. The algorithm runs linearly in the number of aisles. For this end, Ratliff and Rosenthal (1983) define a *partial tour subgraph* (PTS) as a subgraph that spans a subset of the nodes of the graph. When the aisles j are numbered in increasing order starting from the left-most to the right-most, a PTS L_j^- consists of the vertices a_j and b_j as well as all the vertices in aisles up to (but not including) aisle j. PTS L_j^+, on the other hand, consists of all the vertices in aisles up to and including aisle j.

In the algorithm by Ratliff and Rosenthal (1983), one does not have to consider each PTS separately. These can be grouped into *equivalence classes*, which constitute the states of the algorithm. An *equivalence class* is defined by three parameters: the degree parities of a_j and b_j (indicating whether the number of edges incident to the vertex is even or odd) and the number of components in the PTS. The first two parameters can be E (even), U (odd) or 0 (zero), whereas the third can be *0C*, *1C*, or *2C* for no, single, and two connected components, respectively.

Ratliff and Rosenthal (1983) show that there are only seven equivalence classes for any PTS, namely (U,U,1C), (0,E,1C), (E,0,1C), (E,E,1C), (E,E,2C), (0,0,0C), and (0,0,1C). The algorithm starts by forming the L_1^+ configurations for each of the seven equivalence classes, using the six possible connection types described in Figure 3(a). Then, for $j = 2, 3, \ldots, n$, it adds each of the connection types in Figure 3(b) to each L_{j-1}^+ class to obtain L_j^-, which is the minimum time PTS for each class determined by Table 1(a); followed by adding the connection types in Figure 3(a) to each L_j^- class to obtain the L_j^+ classes, determined by Table 1(b). The optimal solution is the shortest of the solutions associated with the equivalence classes (0,E,1C), (E,0,1C), (E,E,1C) and (0,0,1C) for L_n^+.

3.2 *3-OPP*

The algorithm by Ratliff and Rosenthal (1983) can be extended to the k-OPP with $k \geq 3$ by increasing the number of possible equivalence classes. However, in that case, the number of equivalence classes increases exponentially. Roodbergen and de Koster (2001b) provide such an extension for the 3-OPP, which splits the warehouse graph into two blocks. For each aisle, it first enumerates the possible solutions for the lower block, and combines these solutions with the upper solutions after enumeration. The definition of an equivalence class is changed to a quintuplet, where the first four elements are the degrees of a_j, m_{j1}, b_j (which can be E, U, or 0), and the number of connected components in the PTS (which can be *0C*, *1C*, *2C*, or *3C*). The fifth element is applicable only where there are two components with even degree parity. This element shows which two vertices belong to the same component and which one is in the other component of the PTS. These can be *a-mc* (where a_i belongs to one component and m_{j1} and b_i are in the other connected component), *m-ac* or *c-am*.

Under the given settings, the number of possible equivalence classes increases from 7 to 25 and the possible number of connections between aisles increases from 5 to 14. The optimal solution is the shortest of eight candidate equivalence classes. As Pansart, Catusse, and Cambazard (2017) have also shown, as the number of blocks increases, the number of equivalence classes increases exponentially, limiting the use of

Table 1. Transformation of L^+_{j-1} equivalence classes to L^-_j and further to L^+_j equivalence classes, dashed lines indicating suboptimality/infeasibility.

(a) Resulting L^-_j equivalence classes after adding the connection types in Figure 3(b) to each L^+_{j-1} equivalence class

L^+_{j-1} equivalence classes	Inter-aisle connection types in Figure 3(b)				
	(i)	(ii)	(iii)	(iv)	(v)
(U,U,1C)	(U,U,1C)	-	-	-	-
(E,0,1C)	-	(E,0,1C)	-	(E,E,2C)	(0,0,1C)
(0,E,1C)	-	-	(0,E,1C)	(E,E,2C)	(0,0,1C)
(E,E,1C)	-	(E,0,1C)	(0,E,1C)	(E,E,1C)	(0,0,1C)
(E,E,2C)	-	-	-	(E,E,2C)	-
(0,0,0C)	-	-	-	-	(0,0,0C)
(0,0,1C)	-	-	-	-	(0,0,1C)

(b) Resulting L^+_j equivalence classes after adding the connection types in Figure 3(a) to each L^-_j equivalence class

L^-_j equivalence classes	Intra-aisle connection types in Figure 3(a)					
	(1)	(2)	(3)	(4)	(5)	(6)
(U,U,1C)	(E,E,1C)	(U,U,1C)	(U,U,1C)	(U,U,1C)	(U,U,1C)	(U,U,1C)
(E,0,1C)	(U,U,1C)	(E,0,1C)	(E,E,2C)	(E,E,2C)	(E,E,1C)	(E,0,1C)
(0,E,1C)	(U,U,1C)	(E,E,2C)	(0,E,1C)	(E,E,2C)	(E,E,1C)	(0,E,1C)
(E,E,1C)	(U,U,1C)	(E,E,1C)	(E,E,1C)	(E,E,1C)	(E,E,1C)	(E,E,1C)
(E,E,2C)	(U,U,1C)	(E,E,2C)	(E,E,2C)	(E,E,2C)	(E,E,1C)	(E,E,2C)
(0,0,0C)	(U,U,1C)	(E,0,1C)	(0,E,1C)	(E,E,2C)	(E,E,1C)	(0,0,0C)
(0,0,1C)	-	-	-	-	-	(0,0,1C)

dynamic programming-based approaches to only a small number of blocks.

4. Complexity Results

The results in §3 indicate that for the cases of a single block or two blocks, the OPP is polynomially solvable. In this section, we aim to extend these complexity results to a general number of blocks. We start with proving that a more "general" case of the OPP is $\mathcal{NP}$-complete.

Theorem 1. *The OPP is $\mathcal{NP}$-complete for the case where subaisle lengths in the same block and cross aisle segments joining two consecutive pick aisles may be of unequal length.*

Proof. The proof is by reduction from the TSP with rectilinear distances (RTSP), which has been proven to be $\mathcal{NP}$-complete by Garey et al. (1976). Given integer coordinates of vertices in set V on a plane, distances defined by the rectilinear metric, and an integer B, the decision version of the RTSP aims to find a tour that visits all vertices and is of length no more than B.

It is obvious that the OPP is in $\mathcal{NP}$. Assume, without loss of generality, that all vertices have distinct $x-$ and $y-$coordinates, which are sorted as $x_{\{1\}}, x_{\{2\}}, \ldots, x_{\{|V|\}}$ and $y_{\{1\}}, y_{\{2\}}, \ldots, y_{\{|V|\}}$, respectively. The vertex with $y_{\{1\}}$ as $y-$coordinate can be designated as the depot node.

There exist $|V|$ pick aisles, each extending from y_1 to $y_{\{V\}}$ on each of the $x-$coordinate of each vertex. Similarly, $|V|$ cross aisles extend from x_1 to $x_{\{V\}}$ for each $y-$ coordinates of the vertices. This yields an OPP instance with $|V|-1$ blocks where subaisle lengths in block b is of length $y_{\{b+1\}} - y_{\{b\}}$ and the length between pick aisles a and $a+1$ is $x_{\{a+1\}} - x_{\{a\}}$. Figure 4 shows such a transformation of a random RTSP instance with eight vertices to its corresponding OPP instance.

As can also be inferred from Figure 4(b), after the transformation, the items are located close to the back ends of their blocks so that the travel time from that end to the item is negligible. Under such a case, it can easily be shown by inspection that the RTSP instance has a tour with at most length B only if the corresponding OPP instance has one as well. □

Theorem 1 makes use of the fact that the subaisle lengths in the same block and the length of cross aisle segments joining two consecutive pick aisles may not be equal. However, most, if not all instances of the OPP involve equal travel times between two consecutive pick aisles and equal subaisle lengths. In such a case, the transformation in the proof of Theorem 1 ceases to be polynomial. To see this, consider the example in Figure 4(b), and suppose that the the cross-aisle segment lengths starting from the left-most aisle are 6, 18, 5, 3, 16, 5, and 2 units, respectively. Then, to obtain an instance in which the lengths of cross aisle segments joining two consecutive pick aisles are equal, we would need to add 48 empty pick aisles (5 between the first and second aisles, 17 between the second and third, and so on). To generalize this, we would need to find the greatest common divisor of $x_{\{2\}} - x_{\{1\}}, x_{\{3\}} - x_{\{2\}}, \ldots, x_{\{|V|\}} - x_{\{|V|-1\}}$ (for the y-axis, the transformation is identical). Since this depends on the coordinate values, the transformation is no longer polynomial. Thus, the proof is no longer applicable.

To investigate the complexity of the OPP for this specific case, we review the literature on similar problems and present a summary of our findings in Figure 5. Here, thick and thin solid lines represent $\mathcal{NP}-$complete and tractable problems from the literature, respectively. An arrow between two problems indicates that the problem at the root generalizes the one at the tip. Similarly, thick and thin dashed lines represent problems whose $\mathcal{NP}-$completeness and tractability can be immediately deduced, respectively.

The TSP is known to be one of the $\mathcal{NP}-$complete problems (e.g., Garey and Johnson 1979; Johnson and Papadimitriou 1985), many of whose special cases are $\mathcal{NP}-$complete as well, such as the RTSP (Garey, Graham, and Johnson 1976) or with Euclidean distances (Papadimitriou 1977). Since the StSP generalizes the TSP, the former is also $\mathcal{NP}-$complete (Cornuéjols, Fonlupt, and Naddef 1985). A relevant problem is the Steiner Tree Problem (STP), which, instead of finding a tour, aims to find a minimum-length tree that spans a subset of required nodes in a graph. The STP is $\mathcal{NP}-$complete (Karp 1972), as is RSTP, its special case with rectilinear distances (Garey and Johnson 1979). Cornuéjols, Fonlupt, and Naddef (1985) define the graphical TSP (GTSP), in which a node has to be visited at least once in the tour, and show the $\mathcal{NP}-$completeness of the problem, as well as its Steiner version (GStSP), which directly generalizes the OPP.

A *grid graph* is a node-induced subgraph of the infinite graph whose vertices lie on all points of the plane with integer coordinates and whose two vertices are connected only if their distance is 1 (infinite grid). On general grid graphs, the TSP (Itai, Papadimitriou, and Szwarcfiter 1982), the STP, and the RSTP (Garey and Johnson 1979) are all $\mathcal{NP}-$complete, which implies the $\mathcal{NP}-$completeness of the GTSP and GStSP on the same network structure.

A special case of a grid graph is a *solid grid graph*, where there exist no "holes" on the grid. A *rectangular solid grid graph* is a node-induced subgraph of the infinite grid spanning all vertices with integer $x-$ and $y-$coordinates in bounded intervals. The TSP is polynomially solvable on both graph types (Umans and Lenhart 1997), implying the tractability of STP and RSTP on these graphs as well. Note that the OPP generalizes the TSP on solid rectangular grid graphs.

Another important special case is a *series-parallel (S-P) graph*. Two edges are *in series* if they are incident to a node of degree 2 and are in *parallel* if they join the same pair of distinct vertices. An S-P graph is recursively defined as follows: A graph consisting of two vertices joined by two parallel edges is S-P. The graph obtained from an S-P graph by replacing an edge with two series or parallel edges is also S-P. The TSP, GTSP, and STP are all tractable on S-P graphs (Cornuéjols, Fonlupt, and Naddef 1985), as is the GStSP (Baïou and Mahjoub 2002). Since 2-OPP is a special case of the GTSP defined on an S-P graph, this gives another proof for its tractability.

As Figure 5 also shows, no conclusion can be drawn on the complexity of the OPP based on this review. While its special cases of 2-OPP and 3-OPP are polynomially solvable, the OPP only generalizes tractable problems and is generalized by $\mathcal{NP}-$complete problems in our review.

In this section, we have shown that the more general version of the OPP where subaisle lengths in the same block and cross aisle segments joining two consecutive pick aisles may be of unequal length is $\mathcal{NP}$-complete, whereas the complexity of the more specific case where these lengths are equal is open. Consequently, it is quite likely that exact approaches attempting to solve this problem will require substantial computational time as the instance size grows. Indeed, the literature on the OPP has made this assumption inherently, and proposed heuristic approaches. In the next section, we put forward a heuristic approach that makes use of the graph-theoretic properties of the problem.

5. A Graph Theory-Based Heuristic

The main idea of the graph theory-based *merge-and-reach* heuristic for the k-OPP bases itself on the tractability of 2-OPP. The procedure can be summarized as follows. First, a cross aisle is selected as the "cut aisle," dividing the problem into two subproblems. Each subproblem is further divided into its blocks and each of these $k-1$ blocks is solved optimally using the algorithm by Ratliff and Rosenthal (1983). Then, in each subproblem starting from the down-most pair of blocks, we check whether each pair of neighboring solutions overlap (share a common edge or vertex) or not. If they do, the solutions are "merged" by deleting a set of edges from their union without losing connectivity. Otherwise, these disjoint solutions are joined by adding edges, thereby making one "reach" the other. The heuristic proceeds until reaching the upmost block of the subproblem. The last step performs the same procedure for the whole problem, treating solutions of the subproblems instead of solutions of the blocks. The resulting solution is further improved by applying the 3-opt local search procedure. Each middle aisle is used as a cut aisle, and the best solution out of the $k-1$ is reported as a result.

The following definitions are used throughout this section. When two partial neighboring solutions are considered, the lower and upper solutions are denoted as A and B, respectively. A_j and B_j refer to the number of vertical edges (edges that correspond to the pick aisles) incident to the node corresponding to the j^{th} aisle of the lower and upper solutions, respectively. $|A_j|$ and $|B_j|$ denote the number of horizontal edges between the vertices corresponding to the j^{th} and $j+1^{st}$ vertices of the lower and upper solutions, respectively. AB_j simply denotes $(|A_j|, |B_j|)$.

5.1 *Procedure Merge*

The *merge* procedure is called for a pair of overlapping solutions. When two partial solutions overlap, there are two cases that might occur.

In the first case, there are no single vertical edges in any of the two solutions ($A_j \neq 1$ and $B_j \neq 1$ for each j). This can be interpreted as follows: The lower block has a solution that collects all items from its back cross aisle, and the upper block has a solution that collects all the items from its front cross aisle. In this case, the merger is performed by removing a pair of edges on the cross aisle on segments where two pairs of edges overlap. It is easy to show that the resulting solution is the optimal 3-OPP solution on these two blocks. Figure 6(a) illustrates an example for this case. In Figure 6(a), the A_j values are 0, 0, 2, 0, 2, 0, and 2; whereas the B_j values are 2, 0, 0, 2, 0, 2, and 0, respectively. The resulting tour in Figure 6(b) is obtained by keeping these values as they are and using exactly two edges for all cross aisle segments between the left-most and right-most nonempty aisles.

In the second case, there exist single vertical edges in at least one of the partial solutions ($A_j = 1$ and $B_j = 1$), which indicates that some of the items are picked using the front cross aisle of the lower solution and/or the back cross aisle of the upper solution. For the example in Figure 6(c), $A_2 = A_4 = B_1 = B_2 = B_6 = B_7 = 1$. Here, we concentrate our effort on the middle aisle and ensure that none of the vertical edges can be deleted, and aim to minimize the travel time on the middle aisle of the two blocks. Since at most two edges are required to join two vertices, we delete a pair of edges when $AB_j = (2, 1)$ or $AB_j = (1, 2)$. Vertical edges cannot be modified; hence the pairs $AB_j = (0, 1)$ and $AB_j = (1, 0)$ must remain in the final solution to preserve the Eulerian property. Once we delete all the remaining edge pairs on the middle aisle, the

problem becomes one of matching unconnected double vertical edges to connected double vertical edges or single vertical edges. For the example in Figure 6(c), $AB_j = (0,1)$ or $(1,0)$ for $j = 1,2,3,6$. Thus, the resulting merged solution keeps a single horizontal edge for these. Since $AB_4 = (2,2)$ and $AB_5 = (0,2)$, we delete two horizontal edges for these. The resulting merged solution is given in Figure 6(d). In case the resulting solution is disconnected (which does not apply to Figure 6(d)), the subtours are joined using double horizontal edges on the cross aisle.

Algorithm 1 summarizes the merge procedure. The first *if* statement checks whether the first case is satisfied. If so, starting with the leftmost item(s), the picking process uses double vertical edges and ends after picking the rightmost item(s). Otherwise, we delete all AB_j pairs except those with $AB_j = (1,0)$ or $AB_j = (0,1)$, and solve the resulting matching problem. The last *while* loop joins disconnected subtours, if any exist. For this end, we define a counter p and in each iteration of the loop, we join subtours that are p subaisle lengths apart from each other. With n aisles, the complexity of the merge procedure is $O(n^2)$, determined by the last *while* loop: k can be at most n, and we have to search for connectedness of vertical double edges at most n times, ending up with an overall complexity of $O(n^2)$.

Algorithm 1 Procedure *merge*

if $A_j \neq 1$ and $B_j \neq 0$ for all j **then**
  Set $|A_j| = 2$ starting with $argmin_{j \leq n} \left\{ |A_j|, |B_j| : A_j = 2 \text{ or } B_j = 2 \right\}$
  until $argmax_{j \leq n} \left\{ |A_j|, |B_j| : A_j = 2 \text{ or } B_j = 2 \right\}$
else
  $j = 1$
  while $j \leq n$ **do**
    if $AB_j = (2,1)$ or $AB_j = (2,0)$ **then**
      Delete the two edges in $|A_j|$ and set $|A_j| = 0$
    else if $AB_j = (1,2)$ or $AB_j = (0,2)$ **then**
      Delete the two edges in $|B_j|$ and set $|B_j| = 0$
    else if $AB_j = (1,1)$ or $AB_j = (2,2)$ **then**
      Delete all the edges in $|A_j|$ and $|B_j|$, and set $|A_j| = |B_j| = 0$
    end if
    $j \leftarrow j + 1$
  end while
  $p = 1$
  while $p \leq n$ **do**
    Connect all unconnected nodes j with $A_j = 2$ or $B_j = 2$ to j' at distance $|j' - j| = p$ with $A_{j'} = 1$ or $B_{j'} = 1$, or with $A_{j'} = 2$ or $B_{j'} = 2$ connected by horizontal edges to a vertical edge
    $p \leftarrow p + 1$
  end while
end if

5.2 *Procedure Reach*

The *reach* procedure is called when two partial solutions do not overlap, that is, they do not share a common vertex or edge. Unlike the merge procedure, we allow modification of the vertical edges in this situation. Here, we define a *boundary* of an upper (lower) block with the lower (upper) block as the set of downmost (upmost) vertices in each aisle. A boundary of a solution s is denoted by B_s. An *extended boundary* EB_s additionally consists of the first items on the edges corresponding to the aisles adjacent to the boundary vertices. A *portion* P_{v_i,v_j} of an extended boundary with end vertices v_i and v_j is defined such that deletion of the portion as a result of a 2-opt move with the other solution does not break the tour into more than one component.

In the reach procedure, for each vertex of the boundary of the upper solution, we consider extending two edges from this vertex until the end vertices of a portion in the lower solution, and delete the edge between the ends of the portion. In the case where an extension of the two edges creates three edges in total, we delete two of these as well. The vertex-portion pair that makes this move at minimum travel time increase is selected and the reach move is made between this vertex-portion pair. In the case of ties, priority is given to the pair that deletes the longest edge from the portion. In Algorithm 2, the reach procedure is summarized.

Algorithm 2 Procedure *reach*

1: Set $opt = \infty$, $bv^* = v_j^* = v_k^* = \emptyset$
2: **for** each vertex bv_i on the boundary of the upper solution **do**
3: **for** each portion P_{v_i,v_k} of the lower solution **do**
4: **if** $w_{bv_i,v_i} + w_{bv_i,v_k} - W_{v_i,v_k} \leq opt$ **then**
5: $opt = w_{bv_i,v_i} + w_{bv_i,v_k} - W_{v_i,v_k}$
6: $bv^* = bv_i$
7: $v_j^* = v_j$
8: $v_k^* = v_k$
9: **end if**
10: **end for**
11: **end for**
12: Delete edge (v_j^*, v_k^*) and add edges (bv^*, v_j^*) and (bv^*, v_k^*)

The complexity of this procedure is $O(n^3)$, as the boundary of the upper solution can contain $O(n)$ number of vertices for the first for loop, and there can be $O(n^2)$ portions in the lower block for the second loop, ending up with an overall complexity of $O(n^3)$.

As an example, consider the subproblem in Figure 7(a). Here, B_{upper} consists of a single vertex, namely v_5. The boundary B_{lower} consists of m_{14}, m_{24}, m_{34}, v_3, m_{33}, m_{43}, m_{53}, v_4, m_{44}, m_{54}, m_{64} and m_{74}. The extended boundary also consists of v_1 and v_2. Here, the best reach move is found to be from v_5 to the ends of $P_{m_{34},m_{74}}$. Then, we delete a pair of edges from the ones occurring thrice in the intermediate solution, as well as deleting the path between m_{34} and m_{74}. At the end of this procedure, we obtain the solution in Figure 7(b).

5.3 *The Algorithm*

Algorithm 3 summarizes the merge-and-reach heuristic. In each iteration of the outer *while* loop, a cut is generated at the c^{th} middle aisle. After solving the 2-OPP for each block individually, the for loop aims to merge and reach the 2-OPP solutions sequentially for each of the two subproblems separated by the cut. Once this process is over, the algorithm joins the two subproblem solutions using either of the merge or reach procedures. The heuristic then checks whether the solution can be improved, and if applicable, the best solution is updated at the end of the procedure. The best solution is improved by means of a 3-opt local search, which works by removing three edges from the existing solution and reconnecting the network by adding three edges (e.g., Lin 1965; Bellmore and Nemhauser 1968). The reason for choosing this improvement move, despite its computational expense, is due to the fact based on preliminary computational experiments, we observe that the improvement resulting from 2-opt moves is only marginal, if any.

In Algorithm 3, we first start with a cut between the first and second blocks. Here, variable c keeps track of the location of this cut. The whole procedure is repeated for each cut location (c ranging from 1 to $k-2$). First, we apply the Ratliff-Rosenthal algorithm to each block in Line 4. On Line 6, the counters β_1 and β_2 are used to keep track of neighboring blocks. The *while* loop between Lines 9 and 19 first finds the next pair of neighboring solutions (determined once the *if* statement on Line 10 is *false*), and applies procedures *merge* (Line 13) or *reach* (Line 16) depending on whether these solutions overlap or not. Following this, Lines 21 through 25 apply *merge* or *reach* to the two solutions separated by the cut, whereas Line 27 checks whether this is the best solution so far, and updates the incumbent solution, if necessary.

Algorithm 3 Algorithm *merge-and-reach*

$c = 1$, $BEST = \infty$, $BESTSOL = \emptyset$
while $c \leq k - 2$ **do**
  for each block b **do**
    Apply Ratliff-Rosenthal algorithm to the 2-OPP in block b
  end for
  $\beta_1 = c, \beta_2 = k - c - 1$
  for $s = \{1, 2\}$ **do**
    $i = 1, \beta = c(s - 1) + 1$
    while $\beta \leq \beta_s + c - 1$ and $\beta + i \leq c + \beta_2(s - 1)$ **do**
      if $\beta + i^{th}$ block is empty **then**
        $i \leftarrow i + 1$
      else if the β^{th} and $\beta + i^{th}$ 2-OPP solutions overlap **then**
        Apply procedure *merge* to solutions b and $b + i$
        $\beta \leftarrow \beta + i$ and $i \leftarrow 1$
      else if the β^{th} and $\beta + i^{th}$ 2-OPP solutions do not overlap **then**
        Apply procedure *reach* to solutions b and $b + i$
        $\beta \leftarrow \beta + i$ and $i \leftarrow 1$
      end if
    end while
  end for
  if the two solutions separated by cut overlap **then**
    Apply procedure *merge* to solutions s and $s + 1$ and set $i \leftarrow 1$
  else
    Apply procedure *reach* to solutions s and $s + 1$ and set $i \leftarrow 1$
  end if
  Let S be the current solution C be its total travel time
  if $C < BEST$ **then**
    $BEST \leftarrow C$ and $BESTSOL \leftarrow S$
  end if
  $c \leftarrow c + 1$
end while
Improve $BESTSOL$ using the 3-opt heuristic procedure

With $k - 1$ blocks, n pick aisles, and $|I|$ items, the algorithm runs in $O(k^2n^3 + |I|^3)$ time. The first part is determined by the fact that the outer *while* loop (which is executed for $k - 2$ iterations) may call the reach procedure, whose complexity is in $O(n^3)$ a maximum of $k-2$ times in each iteration. The second part is due to the complexity of the 3-opt algorithm. Despite the order of its complexity, the algorithm solves large-size instances in virtually negligible time, as will become apparent in §6.

5.4 *An Example*

For the 4-OPP instance in Figure 1, Figure 8 summarizes the application of the merge-and-reach heuristic. Part (a) illustrates the optimal 2-OPP solutions on each of the three blocks. Regardless of where the cut is placed, all neighboring 2-OPP solutions overlap, hence two iterations of the merge procedure are required, whose result is shown in part (b) of Figure 8. This yields a travel time of 172 units.

For this specific example, the resulting sequence of items is given as $0 \rightarrow 12 \rightarrow 11 \rightarrow 7 \rightarrow 10 \rightarrow 14 \rightarrow 17 \rightarrow 16 \rightarrow 13 \rightarrow 9 \rightarrow 8 \rightarrow 20 \rightarrow 18 \rightarrow 19 \rightarrow 24 \rightarrow 23 \rightarrow 22 \rightarrow 21 \rightarrow 25 \rightarrow 15 \rightarrow 6 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 0$.The first 3-opt iteration replaces node 7 between the nodes 8 and 20. This reduces the total travel

time to 170. The second 3-opt iteration replaces the sequence $13 \rightarrow 9 \rightarrow 8 \rightarrow 7$ between the nodes 15 and 6, reducing the total travel time to 164. The resulting solution, which coincides with the optimal solution, is given in Figure 8(c).

6. Computational Experiments

Our computational experiments have been conducted to assess the performance of the merge-and-reach heuristic in terms of three dimensions: (1) how it compares to the other state-of-the-art heuristics in the OPP literature, (2) how robust it is in terms of changing instance parameters (an aspect found to be lacking for its counterparts), and (3) how the 3-opt improvement affects the travel times of resulting routes. For this end, we first explain the settings used in generating the instances used in the experiments, followed by the computational results.

6.1 *Instance Settings*

For comparability purposes, we follow the settings in Roodbergen and de Koster (2001a) to the best extent possible to generate our instances. Since the specific instances in Roodbergen and de Koster (2001a) are not available, we generate a sufficient number of instances in each setting to ensure that with a probability of 95%, the relative error for average travel time in each setting is within 1% of the mean. For our case, 2,000 instances ensures this for all cases.

The warehouse varies from a single block to 10 blocks. The number of storage locations is kept constant and whenever middle aisles are present, their locations in the layout ensure that all blocks include an equal number of storage locations. Each location is assumed to store a separate item. Uniform demand is assumed, that is, the probability that each item is included in a generated order is equal.

In all settings, the length between two consecutive pick aisles and the width of each cross aisle is 2.5 m. This way, addition of each middle aisle increases the size of the warehouse, as is the case in Roodbergen and de Koster (2001a). The walking speed of each picker is 0.6 m/s. The instances are generated by varying the number of pick items as 10 or 30, length of the aisles as 10 or 30 m, and the number of pick aisles as 7 or 15. With 10 blocks and two levels for each of pick items, number of aisles and aisle length, as well as 2,000 instances for each setting, we solve a total of 160,000 instances using merge-and-reach both without and with 3-opt improvement.

The results are compared to those of the S-shape, largest gap, aisle-by-aisle heuristic by Vaughan and Petersen (1999), and the combined and combined$^+$ heuristics by Roodbergen and de Koster (2001a). We use Concorde TSP solver (Applegate, Bixby, and Chvátal 2011) to find the optimal solution for each instance. If z^* is the optimal length of the picker's tour and z^H is the travel time resulting from heuristic H, we make use of the percent optimality gap, calculated as $\frac{z^H - z^*}{z^*}$, to compare relative performance of the heuristics.

6.2 *Computational Results*

The merge-and-reach heuristic is coded in C++ and executed on a personal computer with Intel Core i7-6600U 2.60 GHz processor with 8 GB RAM. The average run time of each instance is within 0.1 seconds, and hence is not reported. Table 2 presents the percent optimality gaps for (i) the best heuristic result among S-shape, largest gap, aisle-by-aisle, and combined$^+$ heuristics from Roodbergen and de Koster (2001a), (ii) merge-and-reach heuristic without 3-opt improvement, and (iii) merge-and-reach with improvement.

Comparing the performances of the heuristics tested in Roodbergen and de Koster (2001a), one easily observes the dominance of the combined$^+$ (C^+) heuristic over the S-shape, largest gap (LG), aisle-by-aisle (ABA) and combined (C) heuristics. The C^+ heuristic results in the best optimality gaps in 74 out of the 80 instance settings. The S-shape heuristic is never able to obtain the best deviation and the ABA heuristic can find the best average deviation in only one setting. This is because C and C^+ heuristics generalize the

Table 2. Resulting percent optimality gaps for the best heuristic solutions (and the corresponding heuristics) for the instance set of Roodbergen and De Koster (2001b) and average percent deviations of merge-and-reach and merge-and-reach$^+$ for our instance set, bold entries indicating superiority over the best heuristic solutions (2,000 instances for each setting; C^+, LG, or ABA in brackets denotes that the best result is found using the combined$^+$, largest gap, or aisle-by-aisle heuristic, respectively)

	No. of aisles	Aisle length	No. of items	Number of blocks									
				1	2	3	4	5	6	7	8	9	10
Best heuristic result among C$^+$, LG and ABA (Roodbergen and De Koster, 2001b)	7	10	10	5.70 (LG)	2.70 (C$^+$)	3.57 (C$^+$)	3.32 (C$^+$)	3.03 (C$^+$)	2.77 (C$^+$)	2.51 (C$^+$)	2.41 (C$^+$)	2.18 (C$^+$)	1.92 (C$^+$)
	7	10	30	2.95 (C$^+$)	2.40 (C$^+$)	13.14 (C$^+$)	12.02 (C$^+$)	12.90 (C$^+$)	12.58 (C$^+$)	10.94 (C$^+$)	10.57 (C$^+$)	9.41 (C$^+$)	8.52 (C$^+$)
	15	10	10	3.51 (LG)	2.48 (C$^+$)	6.50 (C$^+$)	7.60 (C$^+$)	7.66 (C$^+$)	7.38 (C$^+$)	6.75 (C$^+$)	6.63 (C$^+$)	6.11 (C$^+$)	5.46 (C$^+$)
	15	10	30	5.69 (C$^+$)	2.96 (C$^+$)	18.45(ABA)	20.24 (C$^+$)	23.70 (C$^+$)	24.57 (C$^+$)	22.94 (C$^+$)	22.95 (C$^+$)	21.40 (C$^+$)	19.87 (C$^+$)
	7	30	10	9.46 (LG)	5.61 (C$^+$)	3.88 (C$^+$)	2.82 (C$^+$)	2.27 (C$^+$)	1.85 (C$^+$)	1.76 (C$^+$)	1.50 (C$^+$)	1.41 (C$^+$)	1.33 (C$^+$)
	7	30	30	5.15 (C$^+$)	5.70 (C$^+$)	10.62 (C$^+$)	8.71 (C$^+$)	8.66 (C$^+$)	7.90 (C$^+$)	7.30 (C$^+$)	6.69 (C$^+$)	6.26 (C$^+$)	5.98 (C$^+$)
	15	30	10	6.28 (LG)	4.90 (C$^+$)	4.95 (C$^+$)	4.45 (C$^+$)	4.04 (C$^+$)	3.68 (C$^+$)	3.64 (C$^+$)	3.37 (C$^+$)	3.21 (C$^+$)	3.14 (C$^+$)
	15	30	30	7.53 (LG)	6.84 (C$^+$)	14.46 (C$^+$)	12.51 (C$^+$)	13.78 (C$^+$)	13.52 (C$^+$)	13.01 (C$^+$)	12.67 (C$^+$)	12.18 (C$^+$)	12.00 (C$^+$)
Merge-and-reach without 3-opt improvement	7	10	10	**0.00**	**1.00**	**0.99**	**1.00**	**0.76**	**0.81**	**1.11**	**1.35**	**1.72**	**1.24**
	7	10	30	**0.00**	**0.90**	**0.78**	**1.01**	**1.12**	**1.38**	**1.33**	**1.35**	**1.30**	**1.06**
	15	10	10	**0.00**	**1.01**	**1.75**	**1.45**	**2.05**	**1.97**	**2.14**	**1.69**	**1.49**	**2.01**
	15	10	30	**0.00**	**1.33**	**1.83**	**1.64**	**2.12**	**1.99**	**1.62**	**1.45**	**1.90**	**1.74**
	7	30	10	**0.00**	**1.64**	**1.67**	**2.27**	2.70	2.52	2.03	1.72	2.17	1.72
	7	30	30	**0.00**	**1.41**	**1.50**	**2.00**	**1.79**	**2.31**	**2.25**	**2.03**	**1.98**	**2.21**
	15	30	10	**0.00**	**1.95**	**1.48**	**2.36**	**2.77**	**3.05**	**2.53**	**2.51**	**2.46**	**2.54**
	15	30	30	**0.00**	**1.05**	**2.04**	**1.85**	**2.71**	**2.49**	**2.10**	**2.02**	**2.22**	**2.15**
Merge-and-reach with 3-opt improvement	7	10	10	**0.00**	**0.45**	**0.50**	**0.44**	**0.51**	**0.44**	**0.41**	**0.37**	**0.67**	**0.49**
	7	10	30	**0.00**	**0.56**	**0.56**	**0.48**	**0.68**	**0.81**	**0.50**	**0.79**	**0.73**	**0.68**
	15	10	10	**0.00**	**0.76**	**0.91**	**0.98**	**1.15**	**0.81**	**1.23**	**0.86**	**0.87**	**0.96**
	15	10	30	**0.00**	**0.60**	**0.67**	**0.84**	**1.00**	**0.93**	**1.00**	**1.12**	**0.65**	**0.80**
	7	30	10	**0.00**	**0.85**	**0.82**	**1.10**	**1.18**	**1.02**	**1.08**	**0.83**	**0.74**	**0.77**
	7	30	30	**0.00**	**0.70**	**0.83**	**0.89**	**1.11**	**0.93**	**1.24**	**1.08**	**0.73**	**0.65**
	15	30	10	**0.00**	**0.69**	**0.83**	**1.09**	**1.16**	**1.39**	**1.57**	**0.89**	**0.79**	**0.89**
	15	30	30	**0.00**	**0.59**	**1.04**	**0.99**	**1.29**	**0.85**	**1.07**	**0.95**	**0.99**	**0.84**

moves of the ABA heuristic by allowing multiple entrances to and exits from an aisle. For a single block, all three heuristics result in the same solutions, as C^+ can improve the solutions of C only if there are multiple blocks, and for a single block the combined heuristic allows entering and exiting an aisle only once, as does aisle-by-aisle. The LG heuristic gives the best deviations in five out of 80 instance settings.

There are cases where C and C^+ heuristics fail to find good solutions, specifically under (i) a single block, (ii) larger pick lists, and (iii) longer aisles. For such settings, when the largest gaps occur at the middle of the aisles, the optimal solutions require that the aisle be entered and exited twice, which is not allowed by the C and C^+ heuristics. Under these settings, the optimality gap can increase up to 24.57% (as opposed to a gap of 1.33% gap in the best case). This indicates the instance-dependent performance of the heuristics in the literature, justifying the need for a more robust heuristic.

Merge-and-reach significantly outperforms its counterparts when there exists a single block, in which case it finds the optimal solution using the Ratliff and Rosenthal (1983) algorithm. Over these instances, merge-and-reach improves the best heuristic result by an average of 5.4%, and a maximum of 8.4%. When all settings are considered, merge-and-reach without 3-opt outperform the best heuristic result in 74 out of 80 settings by an average of 6.1% and maximum of 18.2%. Higher deviations occur in settings with pick lists with more items and longer pick aisles. For the remaining settings, the gap between the best heuristic and merge-and-reach without 3-opt is within 0.5%. These are more "sparse" cases with long aisles and fewer pick items, indicating that the reach procedure may require further improvement.

Even without 3-opt improvement, the performance of merge-and-reach is very robust in terms of changes in the instance settings. Disregarding 2-OPP instances, its optimality gap varies between 0.8% and 3.1%, which shows substantially less variation compared to its counterparts. An interesting observation is that while the performance of C^+ initially deteriorates with an increase in the number of blocks and slightly improves when the number of blocks is further increased, the performance of merge-and-reach shows no such correlation with the number of blocks. This can be attributed to the fact that merge and reach procedures work well with dense and sparse warehouse networks, respectively, and hence in both cases, the performance stays fairly stable.

When 3-opt improvements are introduced, there are three main observations. First, we observe that in all 80 instance settings, merge-and-reach outperforms its counterparts, with a rate ranging from 0.5% to 19.0% and averaging 6.4%. Second, the robustness of the heuristic further increases; with optimality gaps ranging from 0.4% to 1.6% (excluding the 2-OPP instances) and averaging 0.8%. Furthermore, the performance of the improved heuristic appears to be independent of any instance setting. Lastly, despite the improvement in robustness and relative performance to other heuristics, applying 3-opt improves our solutions only by an average and maximum of 0.8% and 1.6%, respectively. However, since this improvement is achieved with almost no additional extra computational time, it may be welcome by the management.

7. Conclusions and Further Research Directions

This paper has focused on the order picking problem in parallel aisle warehouses, which addresses the routing of an order picker in response to an upcoming order. The problem has important theoretical and practical implications, as it constitutes a special case of the (Steiner) traveling salesperson problem, and addresses the costliest decision in a warehouse. A more generalized version of the problem has been shown to be $\mathcal{NP}-$complete, whereas for the case of equal-length subaisles and equal travel time between consecutive subaisles, a review of results from the literature on relevant problems have been provided.

Extending the exact algorithm for a single block and making use of the graph structure of the problem, this paper has also proposed a heuristic, which, based on randomly generated instances, has been computationally shown to outperform other state-of-the-art heuristics in the literature in terms of both solution quality and robustness.

During the computational experiments, the proposed heuristic has been compared only to specialized heuristics from the OPP literature. Other general heuristics for the TSP applied on the OPP, such as the $k-$interchange (Makris and Giakoumakis 2003) or Lin-Kernighan-Helsgaun (Theys et al. 2010, where opti-

mality gaps are within 0.4% for all settings), have been left out of consideration, as they ignore the structural properties of the problem.

The work in this paper also points to interesting topics for further research. As indicated by Roodbergen and de Koster (2001a), an immediate extension of the experiments might involve skewed-demand and non-random storage policies. Other warehouse layouts with different middle aisle types, such as the V-shaped and fishbone (Gue and Meller, 2009), as well as chevron, leaf, and butterfly (Öztürkoğlu, Gue, and Meller 2012) are immediate candidates to extend the proposed heuristic.

The paper has assumed that travel time of the order picker is mainly determined by the travel between two items or the depot. In a recent study, Çelik and Süral (2016) incorporate the effect of turns into the travel time calculations and indicate that graph-based heuristics can be modified to handle the number of turns in addition to travel time minimization. The heuristic proposed in this paper is a suitable candidate for such an extension to estimate travel time more accurately.

Despite a detailed literature review on relevant problem, no conclusion on the computational complexity of the specific problem has been achieved, which constitutes another potential research direction. As multiple-picker versions of the OPP are $\mathcal{NP}-$complete, the proposed heuristic may also be used as a subroutine for approaches developed for this version of the problem as well.

References

Altarazi, S. A., and M. M. Ammouri. 2018. "Concurrent manual-order-picking warehouse design: a simulation-based design of experiments approach." *International Journal of Production Research.* To appear. https://doi.org/10.1080/00207543.2017.1421780.

Applegate, D. L., R. E. Bixby, and V. Chvátal. 2011. *The Traveling Salesman Problem: A Computational Study*. Princeton, NJ: Princeton University Press.

Baïou, M., and A. R. Mahjoub. 2002. "The Steiner traveling salesman polytope and related polyhedra." *SIAM Journal of Optimization* 13 (2): 498–507.

Bartholdi, J. J., and S. Hackman. 2016. "Warehouse and Distribution Science. Release 0.97." http://www.isye.gatech.edu/ jjb/wh/book/editions/wh-sci-0.97.pdf. Accessed: July 2017.

Bellmore, M., and G. L. Nemhauser. 1968. "The Traveling Salesman Problem: A survey." *Operations Research* 16 (3): 538–558.

Burkard, R. E., V. G. Deineko, R. Van Dal, J. A. A. Van Der Veen, and G. J. Woeginger. 1998. "Well-solvable special cases of the traveling salesman problem: A survey." *SIAM Review* 40 (3): 496–546.

Çelik, M., and H. Süral. 2016. "Order picking in a parallel-aisle warehouse with turn penalties." *International Journal of Production Research* 54 (14): 4340–4355.

Chabot, T., R. Lahyani, L. C. Coelho, and J. Renaud. 2017. "Order picking problems under weight, fragility and category constraints." *International Journal of Production Research* 55 (21): 6361–6379.

Chackelson, C., A. Errasti, D. Ciprés, and F. Lahoz. 2013. "Evaluating order picking performance trade-offs by configuring main operating strategies in a retail distributor: A design of experiments approach." *International Journal of Production Research* 51 (20): 6097–6109.

Chen, F., H. Wang, C. Qi, and Y. Xie. 2013. "An ant colony optimization routing algorithm for two order pickers with congestion consideration." *Computers & Industrial Engineering* 66 (1): 77–85.

Chen, F., H. Wang, Y. Xie, and C. Qi. 2016. "An ACO-based online routing method for multiple order pickers with congestion consideration in warehouse." *Journal of Intelligent Manufacturing* 27 (2): 389–408.

Chen, F., Y. Wei, and H. Wang. 2017. "A heuristic based batching and assigning method for online customer orders." *Flexible Services and Manufacturing Journal.* https://doi.org/10.1007/s10696–017–9277–7.

Chen, T. L., C. Y. Cheng, Y. Y. Chen, and L. K. Chan. 2015. "An efficient hybrid algorithm for integrated order batching, sequencing and routing problem." *International Journal of Production Economics* 159: 158–167.

Cheng, C. Y., Y. Y. Chen, T. L. Chen, and J. J. W. Yoo. 2015. "Using a hybrid approach based on the particle swarm optimization and ant colony optimization to solve a joint order batching and picker routing problem." *International Journal of Production Economics* 170: 805–814.

Cornuéjols, G., J. Fonlupt, and D. Naddef. 1985. "The traveling salesman problem on a graph and some related integer polyhedra." *Mathematical Programming* 33: 1–27.

De Koster, R., R. Le-Duc, and K. J. Roodbergen. 2007. "Design and control of warehouse order picking: A literature review." *European Journal of Operational Research* 182 (2): 481–501.

De Koster, R., and E. van der Poort. 1998. "Routing orderpickers in a warehouse: a comparison between optimal and heuristic solutions." *IIE Transactions* 30: 469–480.

De Santis, R., R. Montanari, G. Vignali, and E. Bottani. 2018. "An adapted ant colony optimization algorithm for the minimization of the travel distance of pickers in manual warehouses." *European Journal of Operational Research* 267 (1): 120–137.

Dijkstra, A. S., and K. J. Roodbergen. 2017. "Exact route-length formulas and a storage location assignment heuristic for picker-to-parts warehouses." *Transportation Research Part E: Logistics and Transportation Review* 102: 38–59.

Garey, M. R., R. L. Graham, and D. S. Johnson. 1976. "Some NP-complete geometric problems." In *Proceedings of the 8th Annual ACM Symposium on Theory of Computing,* 10–22. New York, NY: Association for Computing Machinery.

Garey, M. R., and D. S. Johnson. 1979. *Computers and Intractability*. New York, NY: W. H. Freeman and Company.

Giannikas, V., W. Lu, B. Robertson, and D. McFarlane. 2017. "An interventionist strategy for warehouse order picking: Evidence from two case studies." *International Journal of Production Economics* 189: 63–76.

Gils, T. Van, K. Ramaekers, A. Caris, and R. B. M. de Koster. 2018. "Designing efficient order picking systems by combining planning problems: State-of-the-art classification and review." *European Journal of Operational Research* 267 (1): 1–15.

Grosse, E. H., C. H. Glock, M. Y. Jaber, and W. P. Neumann. 2015. "Incorporating human factors in order picking planning models: Framework and research opportunities." *International Journal of Production Research* 53 (3): 695–717.

Hall, R. W. 1993. "Distance approximations for routing manual pickers in a warehouse." *IIE Transactions* 25 (4): 76–87.

Helsgaun, K. 2000. "An effective implementation of the Lin-Kernighan traveling salesman heuristic." *European Journal of Operational Research* 126 (1): 106–130.

Henn, S., and V. Schmid. 2013. "Metaheuristics for order batching and sequencing in manual order picking systems." *Computers & Industrial Engineering* 66 (2): 338–351.

Itai, A., C. H. Papadimitriou, and J. L. Szwarcfiter. 1982. "Hamilton paths in grid graphs." *SIAM Journal of Computing* 11 (4): 676–686.

Johnson, D. S., and C. H. Papadimitriou. 1985. "Computational complexity." In *The Traveling Salesman Problem,* edited by E. L. Lawler, J. K. Lenstra, J. K. Rinnooy Kan, and D. B. Shmoys. 37–85. Hoboken, NJ: John Wiley & Sons.

Karp, R. M. 1972. "Reducibility among combinatorial problems." In *Complexity of Computer Computations,* edited by R. E. Miller, J. W. Thatcher, and J. D. Bohlinger. 85–103. Boston, MA: Springer US.

Kou, X., G. Xu, and C. Yi. 2018. "Belt-conveyor based efficient parallel storage system design and travel time model analysis." *International Journal of Production Research.* To appear. https://doi.org/10.1080/00207543.2018.1436784.

Li, J., R. Huang, and J. B. Dai. 2017. "Joint optimisation of order batching and picker routing in the online retailer's warehouse in China." *International Journal of Production Research* 55 (2): 447–461.

Lin, C. C., J. R. Kang, C. C. Hou, and C. Y. Cheng. 2016. "Joint order batching and picker Manhattan routing problem." *Computers & Industrial Engineering* 95: 164–174.

Lin, S. 1965. "Computer solution of the Traveling Salesman Problem." *Bell Systems Technical Journal* 44: 2245–2269.

Makris, P. A., and I. G. Giakoumakis. 2003. "*k*-Interchange heuristic as an optimization procedure for material handling applications." *Applied Mathematical Modelling* 27 (5): 345–358.

Matthews, J., and S. Visagie. 2013. "Order sequencing on a unidirectional cyclical picking line." *European Journal of Operational Research* 231 (1): 79–87.

Matusiak, M., R. de Koster, L. Kroon, and J. Saarinen. 2014. "A fast simulated annealing method for batching precedence-constrained customer orders in a warehouse." *European Journal of Operational Research* 236 (3): 968–977.

Öncan, T. 2015. "MILP formulations and an iterated local search algorithm with tabu thresholding for the order batching problem." *European Journal of Operational Research* 243 (1): 142–155.

Öztürkoğlu, Ö., K. R. Gue, and R. D. Meller. 2012. "Optimal unit-load warehouse designs for single-command operations." *IIE Transactions* 44 (6): 459–475.

Pansart, L., N. Catusse, and H. Cambazard. 2017. "Exact algorithms for the picking problem." *ArXiv:1703.00699* .

Papadimitriou, C. H. 1977. "The Euclidean travelling salesman problem is NP-complete." *Theoretical Computer Sci-*

ence 4 (3): 237–244.

Ratliff, H. D., and A. S. Rosenthal. 1983. "Order-picking in a rectangular warehouse: A solvable case of the traveling salesman problem." *Operations Research* 31 (3): 507–521.

Roodbergen, K. J., and R. de Koster. 2001a. "Routing methods for warehouses with multiple cross aisles." *International Journal of Production Research* 39 (9): 1865–1883.

Roodbergen, K. J., and R. de Koster. 2001b. "Routing order pickers in a warehouse with a middle aisle." *European Journal of Operational Research* 133 (1): 32–43.

Roodbergen, K. J., I. F. A. Vis, and G. D. Taylor. 2015. "Simultaneous determination of warehouse layout and control policies." *International Journal of Production Research* 53 (11): 3306–3326.

Ruberg, Y., and A. Scholz. 2016. *A mathematical programming formulation for the single-picker routing problem in a multi-block layout*. Working paper 2/2016. Faculty of Economics and Management, Universität Magdeburg, Germany.

Scholz, A. 2016. *An exact solution approach for the single-picker routing problem in warehouses with an arbitrary block layout*. Working paper 6/2016. Faculty of Economics and Management, Universität Magdeburg, Germany.

Scholz, A., S. Henn, M. Stuhlmann, and G. Wäscher. 2016. "A new mathematical programming formulation for the single-picker routing problem." *European Journal of Operational Research* 253 (1): 68–84.

Scholz, A., and G. Wäscher. 2017. "Order batching and picker routing in manual order picking systems: The benefits of integrated routing." *Central European Journal of Operations Research* 25 (2): 491–520.

Shqair, M., S. Altarazi, and S. Al-Shihabi. 2014. "A statistical study employing agent-based modeling to estimate the effects of different warehouse parameters on the distance traveled in warehouses." *Simulation Modelling Practice and Theory* 49: 122–135.

Theys, C., O. Bräysy, W. Dullaert, and B. Raa. 2010. "Using a TSP heuristic for routing order pickers in warehouses." *European Journal of Operational Research* 200 (3): 755–763.

Tompkins, J. A., J. A. White, Y. A. Bozer, and J. M. A. Tanchoco. 2010. *Facilities Planning*. Hoboken, NJ: John Wiley & Sons.

Umans, C., and W. Lenhart. 1997. "Hamiltonian cycles in solid grid graphs." In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science,* 496–505. Miami Beach, FL: IEEE.

Valle, C. A., J. E. Beasley, and A. S. da Cunha. 2017. "Optimally solving the joint order batching and picker routing problem." *European Journal of Operational Research* 262 (3): 817–834.

Vaughan, T. S., and C. G. Petersen. 1999. "The effect of warehouse cross aisles on order picking efficiency." *International Journal of Production Research* 37 (4): 881–897.

Wu, Y., C. Zhou, Y. Wu, and X. T. R. Kong. 2017. "Zone merge sequencing in an automated order picking system." *International Journal of Production Research* 55 (21): 6500–6515.

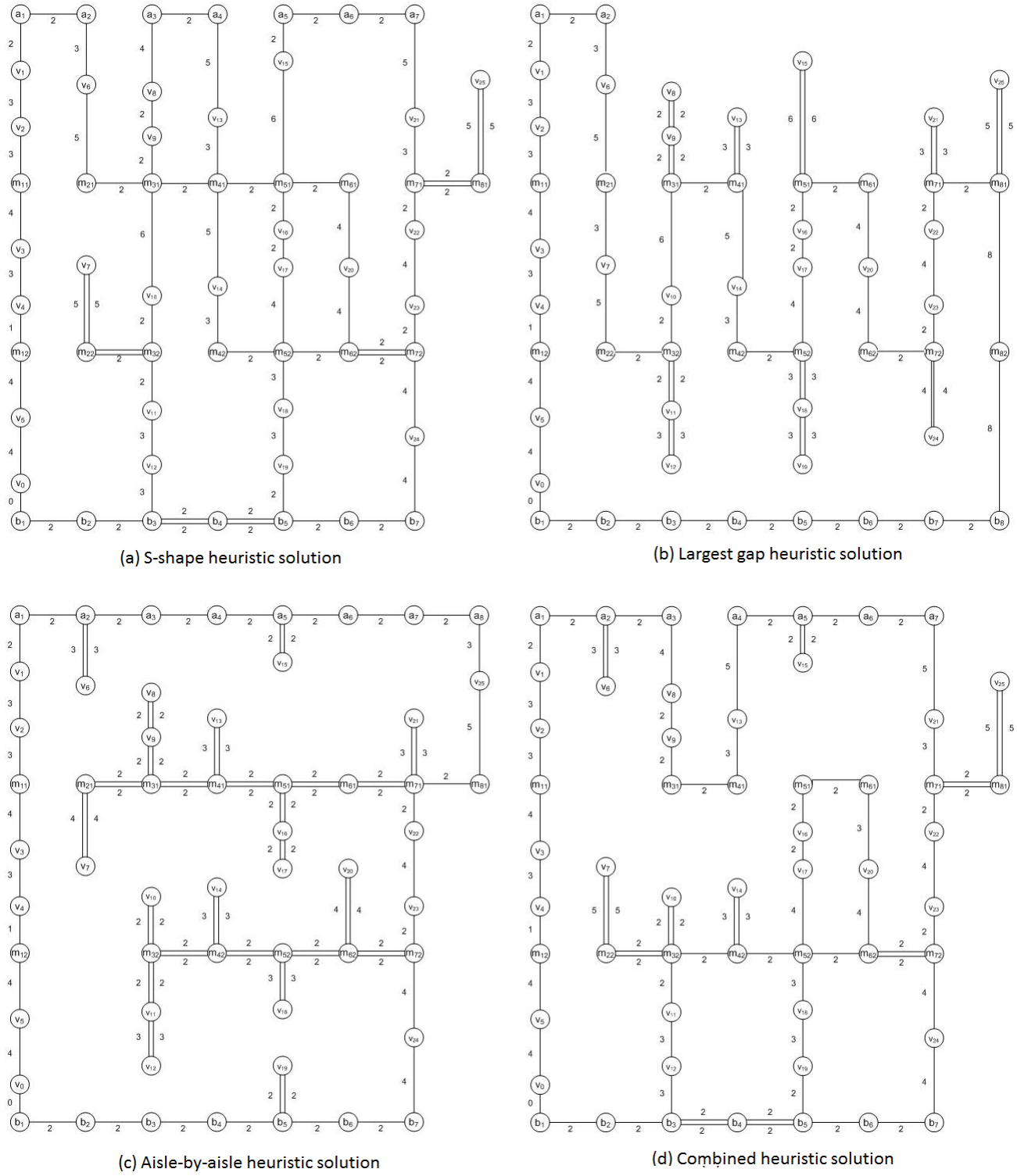


Figure 2. S-shape, largest gap, aisle-by-aisle, and combined heuristic solutions for the instance in Figure 1.

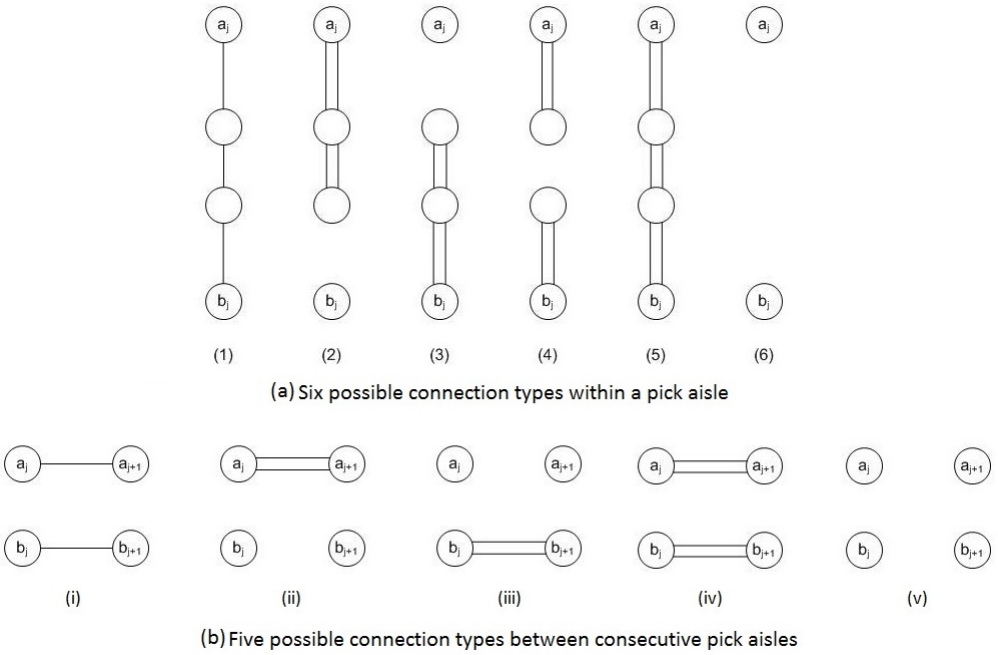


Figure 3. Possible connection types between and within the pick aisles.

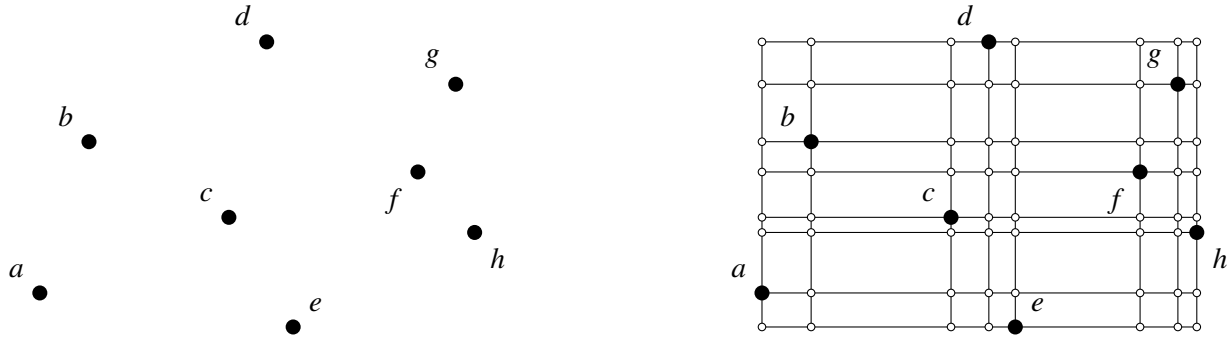


Figure 4. A random rectilinear TSP instance with eight vertices and its corresponding "generalized" OPP instance.

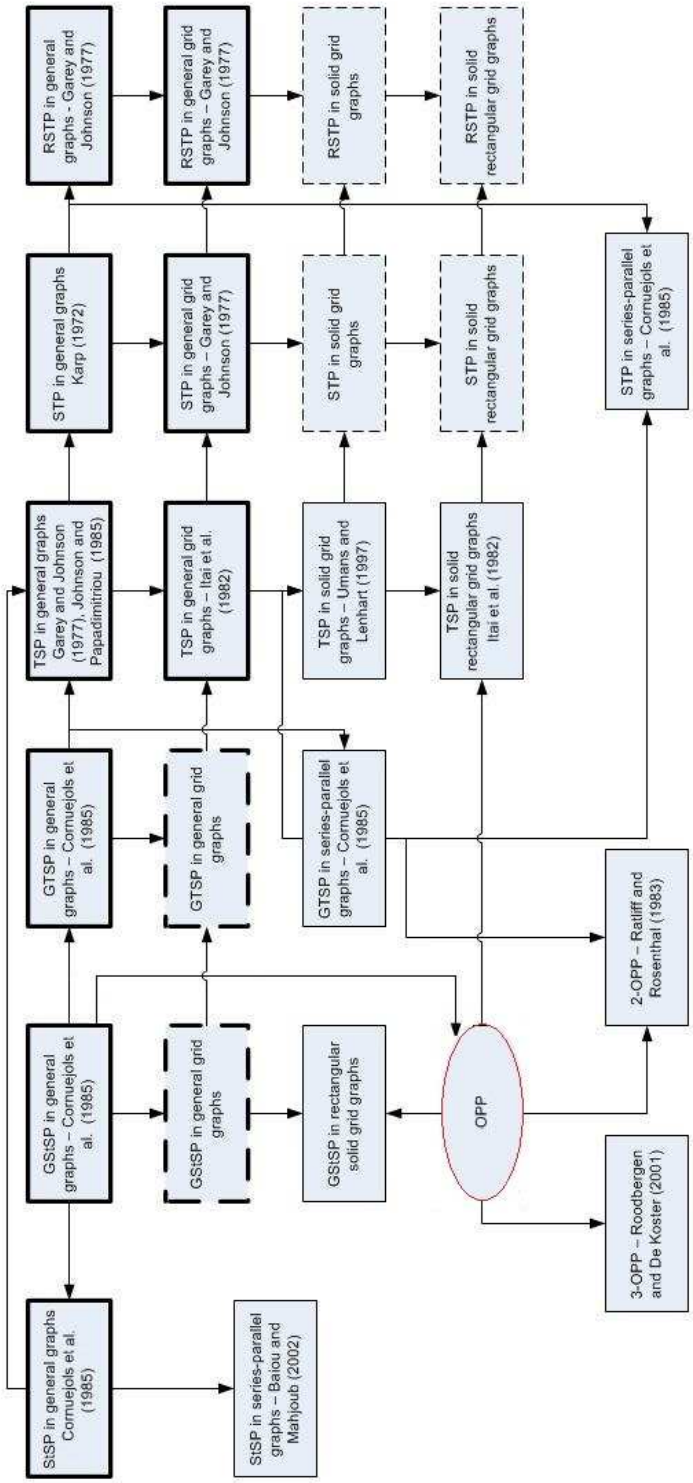


Figure 5. Summary on the complexity of the OPP and related problems.

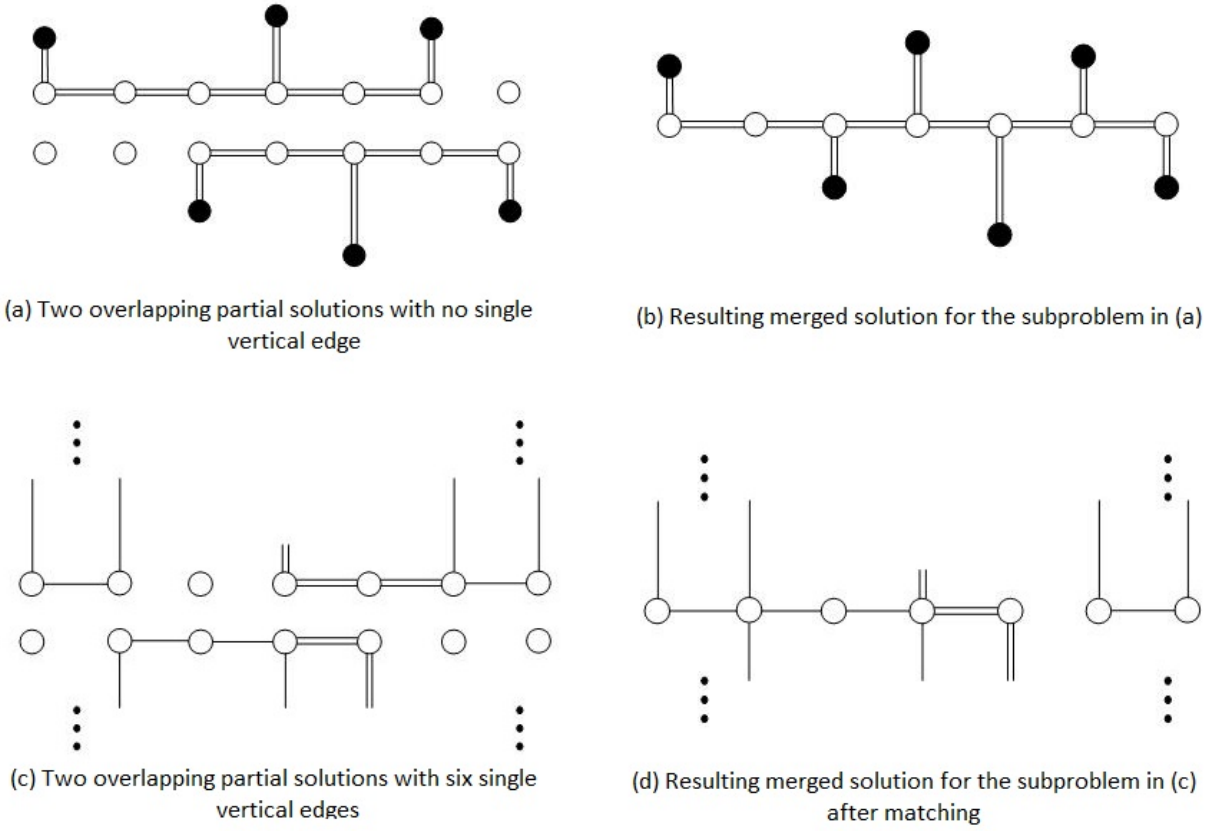


Figure 6. Two cases of the merge procedure based on whether overlapping partial solutions share a single vertical edge, along with corresponding solutions, where solid and unfilled nodes represent items and corners, respectively

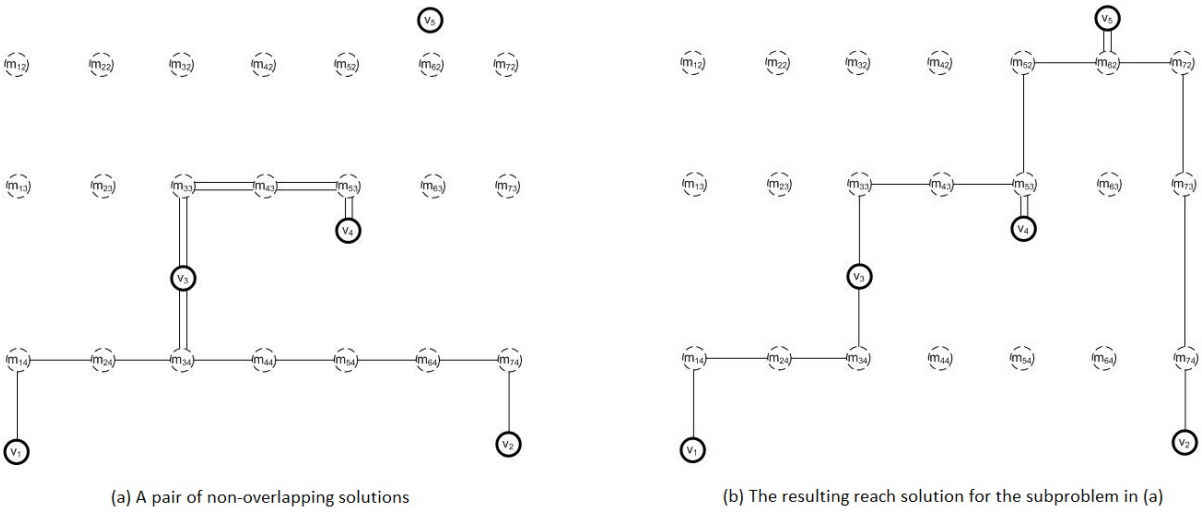
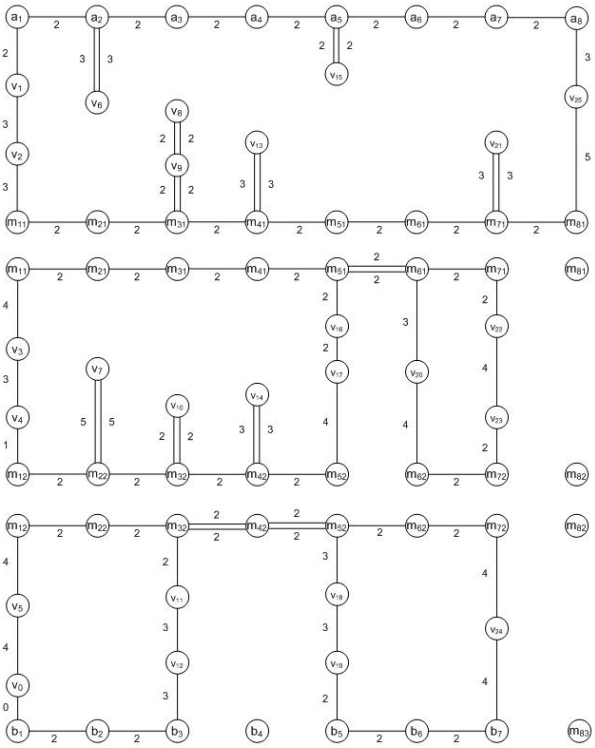
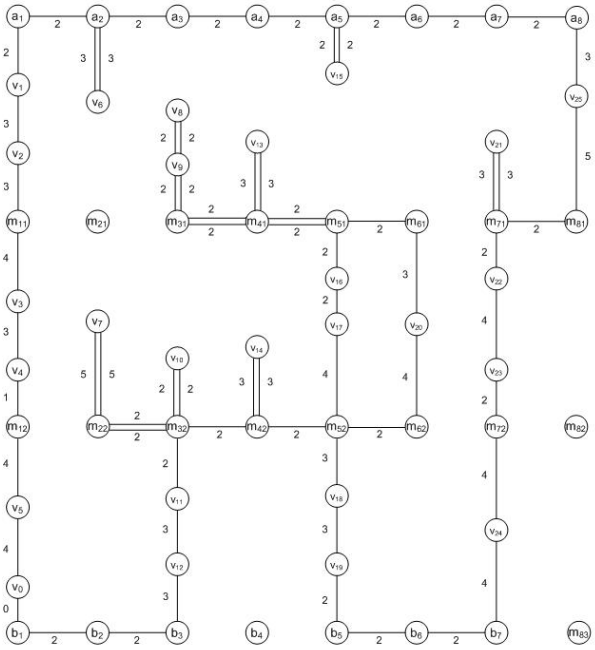


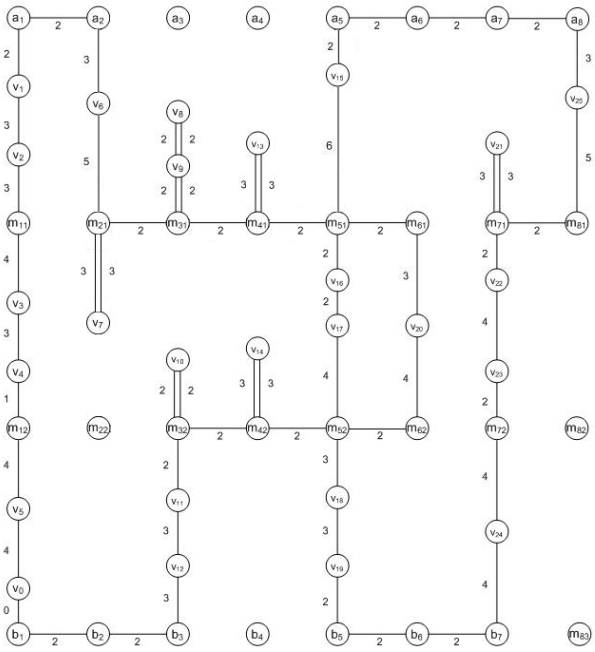
Figure 7. A subproblem where two solutions do not overlap and its corresponding reach solution



(a) 2-OPP solutions for each block



(b) Merge-and-reach solution without improvement



(c) Merge-and-reach solution with 3-opt improvement

Figure 8. 2-OPP and merge-and-reach solutions (without and with 3-opt improvement) for the instance in Figure 1